LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-TR-231812

Force10 P10 Evaluation

John Allen, Robin Goldstone, Shawn Instenes, Bryan Lawver

June 15, 2007

This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

Force10 P10 Evaluation

John Allen
Robin Goldstone
Shawn Instenes
Bryan Lawver
{johna,robing,shawni,lawver1}@llnl.gov

April 3, 2007
FINAL

Acknowledgements

We are grateful to the LLNL ICS data center staff for providing the Force10 P10 devices used in this evaluation. We also thank Scott Campbell of LBL and Steven Carter of ORNL for taking the time to share their P10 experiences with us.

Introduction

The lack of an acceptable intrusion monitoring[1] solution limits the deployment of 10GE (10 Gigabit-per-second Ethernet) technology across the LLNL's unclassified network infrastructure. The desire to operate at 10GE motivates us to evaluate the functionality and performance of a 10GE intrusion monitoring solution, the Force10 P10.

Scope

This evaluation looks at general operational, functionality and performance characteristics of the P10 and also considers the specific use case of the LLNL Green Data Oasis (GDO) project, which is intended to provide LLNL researchers with a means for sharing large-scale datasets with external collaborators. Wherever possible, we have followed recommendations in NIST Special Publication 800-94 *Guide to Intrusion Detection and Prevention (IDP) Systems* for conducting this evaluation. We provide conclusions and recommendations for both the general use case and the specific GDO application.

Physical Inspection

The Force10 P10 is a 1U rackmount PC with dual-core AMD processors running RedHat Linux. A custom network card, known as the MTP (Meta Traffic Processor), is connected via the PCI bus. The MTP-10G is the version of the card designed to inspect 10 gigabit bidirectional (20 gigabit) IPv4 and IPv6 network traffic. The MTP contains a group of Field-Programmable Gate Arrays (FPGA) that are loaded with custom firmware and an embedded processor to do the traffic inspection at full speed. Depending on the rule set provided, a portion of the traffic may be passed across the PCI bus to the CPU of the host system for further processing. The vendor asserts that the P10 can inspect or block traffic at line speed in real time.

The P10 has two XFP slots that accept the three available optics modules including several non Force10 XFPs. The three types of XFPs are LR/SR/CX4. With link supplied to both ports, the MTP module booted up and was ready to pass traffic. Boot time is approximately ninety seconds. The device is intended to go inline in the network to operate in IPS or IDS modes, but can also operate in IDS-only mode via a SPAN (mirror) port. We tested both inline and mirror port configurations

The system as shipped contains a single power supply. From an operational perspective this is less than optimal, especially if the device is deployed in-line. However, the unit appears to be a pretty standard 1U package utilizing a PCI riser card to mount the MTP card. We believe it should be fairly easy to repackage the electronics in a different case

---

[1] The term *intrusion monitoring* is used generically to refer to both Intrusion Detection (IDS) and Intrusion Prevention (IPS) systems. However, the actual design of the Force 10 P10 requires us to evaluate these functions separately. Intrusion Prevention will refer to the ability to recognize and prevent "bad" packets from reaching their intended destination, and Intrusion Detection will refer to the ability to recognize, and alert on the presence of "bad" traffic while relying on outside mechanisms to react to that knowledge.

with dual power supplies if increased reliability is needed.

Theory of Operation

The MTP card implements three kinds of actions for every frame it inspects: BLOCK, IGNORE, and ACCEPT. Block actions cause the MTP to corrupt the frame checksum so that the frame will be discarded by the next hop downstream. Ignore actions cause the packet to be copied correctly but no data about the frame will be passed to the P10 host. Accept actions pass the frame through the PCI bus and device driver to the host CPU where it can be processed by any application running on host. Accept frames are also passed to the other network port and on across the network. The MTP firmware that is built by the included compiler orders the rules such that block rules have precedence over ignore rules, which have precedence over accept rules. As we will see later, compilation of rules is a major limitation of this system. Ordering of rules minimizes the data passed to the host CPU. This is important because the MTP cannot send nearly as much data up to the host CPU as it can block or ignore. This is discussed further in the *Technology Limitations* section below.

The MTP supports two different types of rules: static and dynamic. Static rules are compiled into the FPGA firmware. Dynamic rules utilize both the FPGA and the coprocessor without compilation. Once a packet makes it to the host CPU, the packet appears to the host to have come from a "regular" NIC in promiscuous mode and can easily be processed by any Linux-based network security tool such as tcpdump, Snort or Bro etc. A complier is provided to convert Snort style rules into the MTP firmware, allowing some Snort rules to be instantiated in the MTP card, where they can cause traffic to be blocked, and also allowing Snort to run on the host CPU, where it can report in the expected ways on the subset of traffic sent to it.

Security Capabilities

Physically, the card's FPGAs and CPU have access to the full contents of each network packet. The host CPU can normally access the full network packets, though under heavy load conditions some truncation can occur (see Lab Testing section below for more information). The included software understands both IPv4 and IPv6 and the supplied tools only work with those types of packets. If one wanted to write one's own firmware in Verilog, one could access the full Ethernet frame such as would be necessary for VLAN decoding etc. The ability to keep state on the FPGAs and coprocessor is very limited. Snort normally uses pre-processors to avoid many obfuscation attacks allowing the Snort rules to contain only the most readable encoding type. Such pre-processors are not available on the card and only to a subset of data passes to the host OS. For this subset of traffic, all the normal capabilities of Linux running Snort, or any other existing Linux IDS are available. The device can inspect and block data at full rate, but it cannot report on packets it blocks. Blocked packets never reach the host CPU where reporting is done. The P10 can fully report on the subset of data that reaches the host CPU, and if alert data is a sufficiently small percentage of overall traffic, it can apply fully stateful IDS inspection and alerting to that data.

The very high throughput of the P10 for actions that can be completed on the card makes

it a very stealthy device on the network. It utilizes no MAC address, no IP, and induces so little latency that it would be very difficult for an attacker to detect on the network. If the P10 host OS were compromised, it is still be very difficult for it to send data out the monitoring interface due to the fact that it doesn't have a MAC address, and custom Verilog programming would be necessary to do anything other than pass or deny traffic passing though the MTP card. However, a compromised P10 could permit a DoS.

Technology Limitations

Encoded traffic is particularly challenging for this device. The limited capacity of the FPGAs requires the host CPU to evaluate encoded traffic in IDS mode. Encoded traffic in IPS mode is virtually impossible.

To the extent that the P10 completes a decision within the MTP card, it operates at full 10Gbps speeds without packet drop. However, if the packet must be passed to the host CPU, the rate of processing is greatly reduced. We measured a rate of 483-700Mbps depending on the packet size. When the data rate exceeded this PCI card limit, the MTP truncated the packet as it was forwarded to the host. Packets exceeding 128 bytes were truncated to 128 bytes. A detailed analysis of this packet truncation issue is described in the "Lab Testing" section below. Here we will simply note that ACCEPT processing has the potential to trigger truncated or dropped packets. However, by programming a ruleset that limits the amount of traffic passed to the host CPU, this situation can be minimized or avoided entirely.

This system has two types of rules, static and dynamic, as mentioned above. Changing dynamic rules is easy, but since they take up ~10X the space of static rules, realistically it is only possible to make use of a very small number of dynamic rules. We also note that there are significant limitations on the types of constructs that can be expressed with dynamic rules. Basically, they can only match data within IP/TCP/UDP/ICMP headers. This further limits the usefulness of dynamic rules.

Static rules require a complete recompile of the firmware. In our testing, a compilation with close to the maximum sized rule set took in excess of 6 days. The compilation time is dependent on the complexity of the rules and hence cannot be easily bounded. Similarly, the maximum number of rules is data dependant, which means dropping one rule to add another may fail.

The static rule set compilation process is time consuming, error prone and unpredictable. This is clearly a major limitation of the P10. However, if rule changes are infrequent and small in number, we can possibly live with this limitation by using dynamic rules for immediate updates and then rebuilding the static rule set as a background task. This strategy works as long as the recompile completes before the next update is required.

The inability to report specific information on traffic that is blocked is another troublesome limitation of this device. Silent unrecorded packet drops can be mistaken for transmission loss, resulting in a complex debugging scenario. We do note that it is possible to extract counts of packets blocked from the card if necessary.

Prevention Capability

If one has a mature prevention rule set in Snort format that doesn't depend on Snort pre-processors, this device can monitor traffic at full 10Gbps speeds and can drop up to 100% of traffic as dictated by the rules. This is an extraordinary capability but comes with the limitation of not being able to report that it is blocking the traffic. As previously mentioned, the block action consists of corrupting the packet checksum so it is possible that one could keep track of blocks by monitoring the downstream device for packets dropped due to CRC error. However, this would be presuming functionality of a network component that is not part of the P10 solution. Without the ability to report on blocked traffic, the P10 is not a very useful IPS solution. In theory, the P10 could work as an inline firewall if all of the rules fit into the FPGAs, but that usage model is outside the scope of this evaluation.

Usability / User interface

The native interface provided with the P10 is basically a Linux CLI. However it was trivial to install the Sourcefire Defense Center client for Linux. With the Sourcefire client installed, we were able to forward alerts to a Sourcefire aggregator. This provides a more robust monitoring and reporting capability. However, the Sourcefire client does not provide an interface to the rule set compilation process. Compiling the MTP rules is an entirely separate, manual process, with the limitations described above.

Snort rules are mostly accommodated, with previously mentioned restrictions on the use of pre-processors and other highly state driven features. With the caveat that blocks (IPS mode) are not reported at all, Snort alerts on traffic passed to the host CPU can be reported in a standard fashion. Once data gets off of the MTP card, any (RedHat) Linux tool such as open source Snort or Bro can be used to generate alerts.

The P10 fails closed with host OS crashes but fails open with hardware failures such as power supply. Hardware failures can be mitigated with an external optical bypass switch.

Feedback from other DOE sites

As part of our investigation, we arranged two telecons with other DOE sites that have used the Force10 P10, in order to understand how they are using this device in their environments, and to compare notes on our experiences. We spoke to Scott Campbell from LBL and Steven Carter from ORNL. Summaries of these conversations follow.

Scott Campbell, LBL

Scott was using the P10 is a somewhat different and very interesting way. His project involved using the P10 as a front-end "blunt filter" for dynamic protocol analysis. He used a very clever feature of the P10 to send the first six packets of a session up to the host OS, allowing for protocol analysis on a large number of ports for all sessions. Data of interest was then passed along to a system running Bro for more detailed analysis.

Due to the nature of Scott's project, he had no need or opportunity to evaluate the P10 in terms of maximum processing capability or handling of large rule sets. In that regard, he did not run into some of the limitations with the P10 that we have documented in this paper.

Another aspect of Scott's project was doing content matching looking for HTTP traffic on any port. This was done using a set of rules created for SC 2007 by Livio Ricciulli, the designer of the P10. Scott sent us this rule set in case we were interested in exploring the use of the P10 for this kind of analysis.

Though Scott had a very different usage model for the P10 in his project, his overall assessment of the device was very similar to ours, namely:

- It can serve as a very powerful IDS if your specific use case meshes with what the device does well.
- It is not useful as a general purpose IPS and in fact, he would not consider using the device in-line due to concerns about reliability of the hardware.

Steven Carter, ORNL

Oak Ridge is preparing to move their HPC computer facility out from behind the ORNL internal firewall some time in April. At that point, they will have no traditional firewall but will instead use Cisco ACLs and the P10 in IDS mode. Rather than attempt to use the P10 as an IPS, they will use a secondary device to perform active session termination after the packet goes past the P10.

ORNL does not actually have the Force10 P10 product but rather they are using two Meta-Network cards packaged in a high-end Opteron system, providing coverage for dual 10GE up-links to their data center. They use Snort, and subscribe to SourceFire's commercial alert updates. They have developed scripts that select some alerts for compilation into the MTP card, and some alerts to run in Snort on the host. They do the division by port number. We find this to be a very interesting idea, if not for GDO then perhaps for LC's future 10GE connection. Steven agreed to send us a copy of his Perl scripts to do this rule distribution so that we may consider how to make use of this approach in the future.

We also learned from Steven that there is a newer firmware version (1.4.3) that offers some additional features and ability to keep more state information than the 1.4.2 version we have. The increased state information potentially allows the P10 to provide firewall-like capability. The version 1.4.3 MTP compiler also appears to have performance improvements to address the long compilation times that we observed. We will be contacting Force10 to pursue obtaining this newer firmware.

Steven mentioned that Force10 is working on a 2.0 version of the MTP card that was initially supposed to be available last fall, but has been delayed. The new card uses PCI-X and adds more FPGA space. While PCI-X is an improvement over the current PCI implementation, we would have preferred to hear that they were going to use PCI-E.

Steven also discussed another vendor that a 10GE NIC with programmable logic that could be used to do things like send only certain packets or parts of packets to the host OS. Bryan Lawver is familiar with the vendor and is going to pursue getting literature about this product.

Finally Steven mentioned an presentation by LBNL where they described getting around the P10's PCI limitation by using the device as a filter to drop traffic, and then using a conventional monitor on the 10GE port as a replacement for the slower PCI bus as a way to pass on more traffic. It is apparent that most of us who have attempted to use the P10 have run into similar limitations and a lot of creative intelligence is being applied to finding ways to get the most benefit out of this device.

Steven Carter had very much the same overall evaluation of the P10 as Scott Campbell. He agreed that the device is not general purpose, but could be made to work in certain environments. In ORNL's case, they have determined that the P10 can meet their needs as an IDS for protecting their HPC data center. Steven would like to keep in contact with anyone in DOE that actually uses these systems in production.

Lab Testing

The P10 was subjected to an extensive evaluation in LC's network testbed. Figure 1 illustrates the setup that was used for this evaluation.

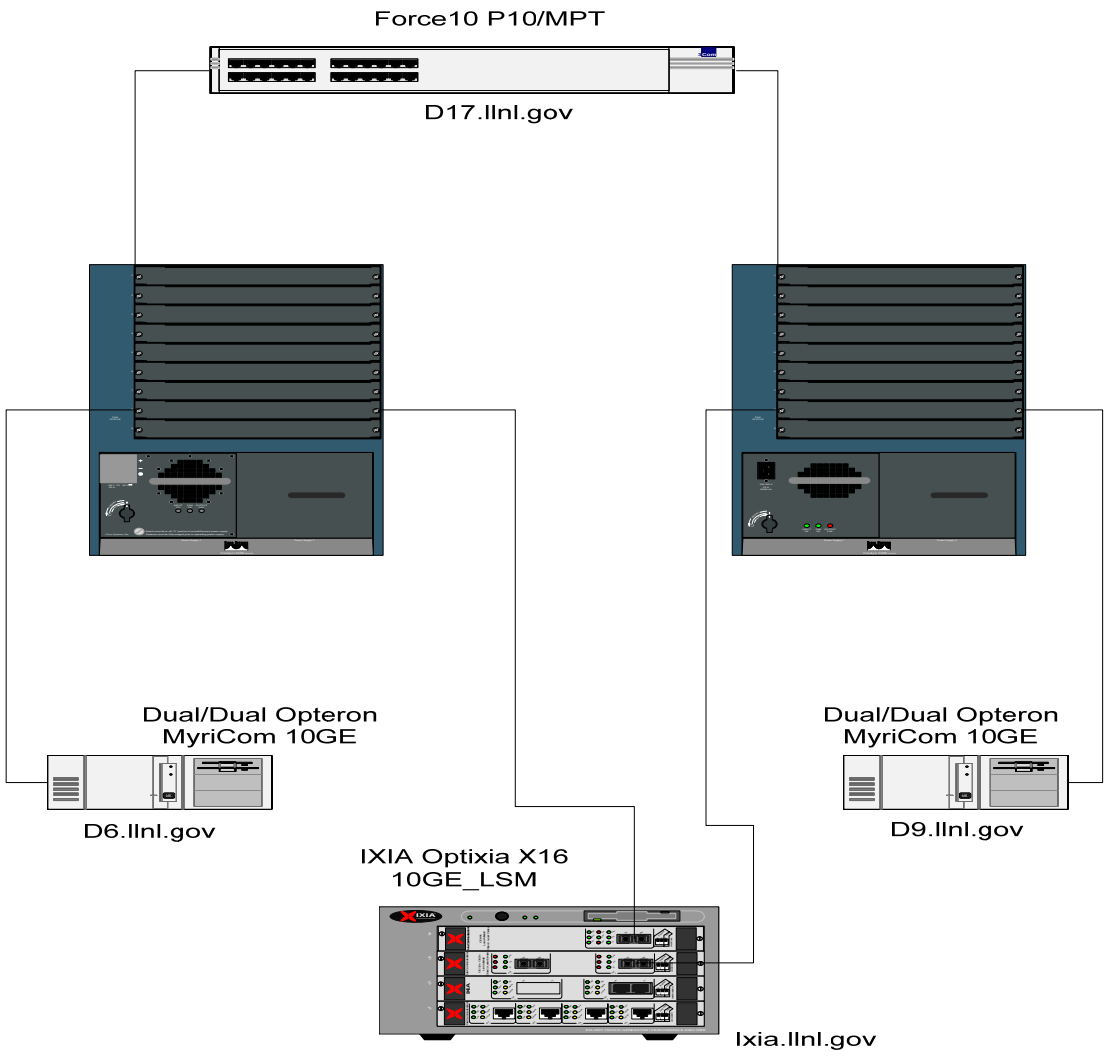

Figure 1: P10 Network Testbed Configuration

The testbed evaluation was focused on determining how well the P10 integrates into an existing network and how much traffic the device will forward. Two 10GE capable Cisco 6509s connected the P10 to test equipment rather than connecting directly to the P10. The Cisco switches were studied during the BlueGene-L federated core installation and found to not bias this type of testing. Two types of 10GE test devices were connected: an Ixia test system and two dual-processor dual-core Opteron hosts with Myricom 10GE NICs running Linux. The Ixia device is capable of generating a very large range of traffic types including huge ranges of addresses and thus is useful for simulating a large network of traffic. The Opteron test hosts can create a single session with bandwidth up to one GigaByte per second. The difference between these two test devices is the Ixia doesn't need to obey TCP congestion rules so it can generate steady

level of traffic irrespective of the packet loss. The test hosts obey TCP congestion rules and as such will drop to a very low throughput in the face of high packet loss.

Latency test

The Ixia has a latency test where it tags the packets with time codes and measures send and receive times. It appears to have a 1ns clock. Latency of the 6509s was measured with and without the P10 in line. The difference was 0.5us, which is at the low end of the advertised range of 0.4-1.3us. We note that a relatively small rule set was used for this testing and that a larger rule set could increase the observed latency. However, based on test results and the design of the device, we believe that the advertised latency range is valid.

Traffic tests

The Ixia generates traffic irrespective of congestion. Various Ixia tests at full bandwidth indicated that this device was near 10Gbps line rate without many dropped packets. These tests included large distribution of MAC and/or IP addresses.

A second set of tests combined the Ixia and Netperf running on the Linux hosts. Netperf is a tool to measure high data rates while obeying TCP congestion rules. It sends traffic to a server which ACKs the traffic and when the test is complete it will report the utilization of the remote end. We verified that the hosts/NICs can reach about 8Gbps and the P10 does not affect this performance. A second test variation started with 50% traffic using the Ixia and then measuring the Netperf rate with this base traffic level. In this case, Netperf report 49.5% throughput which would mean that the P10 was passing 99.5% of the packets in the Netperf direction. The fact that Netperf reached 100% of the remaining available bandwidth is significant. Long before a pipe reaches 99% utilization, other things typically break. So this test indicates that the P10 will not affect flow.

Packet truncation investigation

In a lab test designed to fully exercise the data path through the MTP to the host OS, it was observed that packets were being truncated to 128 bytes under what we will generically call “high load” conditions. This behavior was not well understood initially but after further investigation and analysis of the P10 documentation, we can offer the following explanation for when and why packet truncation occurs.

When an ACCEPT rule on the MTP card matches and data is moved from the MTP card to the host OS, there is processing load placed upon the PCI bus and MTP driver. This maximum load that the P10 can handle (report on) is a combination of interrupt handling speed, PCI bus speed, driver efficiency, host CPU and possibly other factors, which we will generally refer to as the P10’s *processing capacity*.

The P10 has two ways to attempt to deal with high load: truncating packets sent to the OS or dropping them entirely. Initially it will attempt to send shortened versions of the full packets, but as load increases matched packets will be dropped by the driver and will not be seen at all by the host OS. We observed discontinuity in the load vs. truncated

rate graph where under some levels of load the truncation rate will drop briefly as the driver begins to discard packets. Under even higher load (9 Gbps of 64-byte packets all of interest) the driver reports error rates on the interface as well. Performance at this load level is very poor; very few packets make it to the host OS at all.

We note that this behavior can actually be considered an improvement over what a typical sensor would do, which is to drop the packet entirely under high load. In this context, the P10 could be deemed to be making a "best effort" to deliver information about events under high load. In the worst case, the host OS will see a stream of 128-byte packet fragments for matches within the header. The P10 appears to be programmed to attempt to deliver maximum information about as many reportable events as it can, and will favor partial packets that still match the rules over delivering full packets if it must choose between the two. As long as the rules and traffic are set such that the data sent to the host OS from the MTP is within the processing capacity of the device driver and PCI bus (roughly 500Mbit/sec), packets are rarely truncated.

Jumbo frames
The NetPerf and Ixia traffic tests were conducted with 9000 byte Jumbo frames. This lowers the packet rate while maximizing the bit flow rate.

MTP
This tool runs on the P10 and shows packet/byte rates on the three interfaces. This primitive tool was useful while running the Ixia and Netperf tests. The packet rates that the MTP reported matched the rates reported by the Ixia and Netperf.

Port mirroring
A separate test (i.e. not part of this evaluation) of mirror ports with jumbo packet size at 10GE data rates shows that at least one type of (Cisco) switch did not maintain a sequential ordering of send and receive packets. The small ACK packets showed up much earlier than the data packet they were acknowledging. It is expected, but not demonstrated, that out of order packets could also occur using the standard MTU size. However, given that stateful IDS tools such as Snort are able to reconstruct state, including dealing with out of order packets, we do not believe this is a significant issue and thus the mirror port configuration is acceptable for the IDS mode of operation.

Conclusions

In this section we will present our conclusions regarding the use of the P10 for both the GDO 10GE application and as a general-purpose IPS for use within the Lab's backbone network.

GDO

We believe the P10 can serve as an acceptable solution for the GDO if one makes the following assumptions.

1. IDS at 10Gbps is necessary but not IPS.
2. IDS rules only for services/ports that GDO uses is sufficient and it is ok to not notice attacks that are doomed to fail against GDO because the service doesn't exist.
3. Sourcefire Defense Center allows IMT to monitor the P10 in a cost effective manner, consistent with their existing IDS solution.
4. Rule changes will be infrequent and skilled staff is available to implement them.
5. Emergency rule additions can be limited to two rules in each direction.
6. A 10GE SPAN port is available on the GDO switch so that the P10 does not need to be deployed inline.

With these simplifying assumptions, we built a rule set that covers the small set of services/ports that are available on the GDO, and still allows for 4 dynamic rules (2 in each direction). This provides the ability to not only monitor attacks to/from the GDO, but also to insert dynamic rules quickly without a lengthy recompile.

General purpose IPS

The P10 is designed to allow the host CPU to perform stateful IDS on a subset of the 10Gbps of data that is passed through the device. However, given the significant reduction in processing rate for data that must be passed up to the host (483-700Mbps depending on packet size), the usefulness of the P10 solution depends on how much data the MTP can ignore and not pass to the host CPU for processing. So the answer is really data dependent. While this has been true of most IDS systems, Gigabit IDS with which we are familiar usually can look at a larger percentage of the data than can the P10 at this time. On the other hand, most first generation 10GE IDS installations are likely to be running well below full saturation most of the time. The key is whether or not one can express a rule set in the FPGAs that ignores most traffic, such that what is passed up is within the capability of the card to provide to the host CPU for full featured processing. Typically, the deployers of IDS solutions are not accustomed to needing to plan capacity to the degree that is required for this device.

Operationally, the inability to get information about blocks and the complexity of updating the rule set are significant obstacles to a more general-purpose deployment of the P10. In order to support being used in IPS mode, for example, in the main path to the Internet, we need to not only block at full speed, but know if/when a block occurs. The P10 device has no ability beyond incrementing a counter to notify the host system of a block and no way to report any information about the blocked packet that would help determine what was blocked and why, if one had more than one block rule. The process

to update rules is complex, and how to test rules with no notification is a big question. Other issues like being inline in the network, but having only a single power supply are also a concern, but are insignificant compared to others.

In summary, we do not believe the P10 is a “General Purpose” IDS as of yet. Improvements to the rule compilation process would be highly desirable. A faster MTP card (e.g. PCI-Express) to allow more traffic to be sent up to the host would significantly improve this device as a high speed IDS solution and would mitigate the worst of the issues. Ideally the device should also include a fourth action type i.e. “analyze” which would not pass packets on across the network until they can be analyzed. There would obviously be a latency penalty for this operation, but combined with a faster MTP card, this would provide a credible IPS capability.